GROMA
Programming a Permanent Multichannel Sound Installation

Martin RUMORI
Academy of Media Arts Cologne
Peter-Welter-Platz 2
50676 Köln
Germany
rumori@khm.de

Abstract

In this paper I shall be addressing some of the more complex issues of developing the technical framework for the permanent urban sound installation *GROMA*, which incorporates environmental sounds from two partner cities of Cologne, Rotterdam and Liège, combined with ancient texts about urbanisation. This installation has been inaugurated in 2008 at the location of Europe's largest underground parking lot in Cologne. The programming environment Supercollider was used for realising the algorithmic rules of the score and a flexible routing scheme for multichannel sound distribution.

Keywords

sound installation, algorithmic composition, Supercollider, multichannel sound projection

Figure 1: View from staircase *TR 5.02*

1 Introduction

GROMA [1] is a permanent urban sound installation by Cologne based artists Michael Scholz, Judith Nordbrock, and the author of this paper. This multichannel work has been inscribed into two staircases of an underground parking lot, which is situated in the area of *Rheinauhafen* [2], Cologne's former city harbour at the river Rhine. It is now the last big urban development area in the centre of Cologne. The sound installation therefore is intrinsically tied to this new *Rheinauhafen* neighbourhood. The parking lot consists of only one level, which is entirely subterraneous, but through its length of approximately 1500 meters it is currently the longest one in Europe (Figure 2).

The title of the installation, *GROMA*, is derived from the ancient Roman instrument for topographical survey, Groma. It was used by the so called *agrimensors* for determining the basic orientation of a new settlement.

In *GROMA*, sound situations from two partner cities of Cologne, Rotterdam and Liège, are combined with ancient texts about urbanisation in an algorithmic multichannel composition. Each staircase stands for one of the partner cities. Texts used in *GROMA* include excerpts from *Ten Books on Architecture* of Roman writer, architect and engineer *Vitruvius*. Other texts are from Ancient Greece or Ancient Near East and date back up to the 4th millenium before common era. All together they form a canon of topics which are still relevant to today's urban structures and current development.

The two staircases into which the installation is inscribed connect the underground parking lot with the above ground area. They are situated at the two main squares of the new neighbourhood with a distance of approximately 500 meters in between them, and therefore form the two main entrances of the parking lot, which has 26 staircases in total.

All the staircases are spaces which are normally only experienced in transition, and they are by no means comfortable places for a concentrated listening situation. *GROMA* focuses on this volatile moment by providing an artificial third level between the beneath and above, an eternal algorithmic sound composition which can never be experienced in its entirety.

Figure 2: Parking lot underneath *Rheinauhafen*

Figure 3: Speaker designed for *GROMA*

2 Technical Setup

2.1 Speakers

When developing the concept for the *GROMA* sound installation, it quickly became clear that the experience of passing through the staircases should be emphasised by a multichannel sound projection, which would allow sounds to travel, but also to appear from different perspectives towards the building and the passerby. The bigger staircase numbered *TR 5.02* was equipped with twelve speakers and plays the sounds from the city of Rotterdam. The speakers cover the building up to the above ground visitors' platform which provides a nice view over the city of Cologne (Figure 1). The smaller staircase numbered *TR 7.02*, playing the sounds of Liège, does not have such a platform, therefore eight speakers are sufficient.

Upon request of the architects of the parking lot, the speakers had to be as inconspicuous as possible while providing reliability and stability against weather and vandalism attempts. Therefore, especially designed speakers were used for GROMA (Figure 3), which were built by Johannes Heppenheimer from Audiance [3].

2.2 Audio System

The simplest and most reliable way of driving two installations of twelve and eight channels respectively with an algorithmic, global, thus distributed score was to use two separate networked audio systems. They consist of a rack containing one standard computer with a RME multichannel sound card, ADAT to analogue audio converters and multichannel amplifiers.

The parking lot is constructed for being flooded completely in case of an extremely high water of the river Rhine, because otherwise the fundament of the building would be raised by water pressure. Therefore, the racks of the audio systems are on wheels and the complete wiring is realised with removable plugs.

Currently, two measurement microphones are being installed in each staircase which will allow for slight adjustments to the overall volume of the installation according to the surrounding mean sound pressure level.

The computers are running a stable version of the 64 Studio [4] Linux audio distribution. The installation is controlled entirely by the Supercollider [5] sound environment.

2.3 Network

The network connection for synchronising the two computers was realised with an optical connection. Copper cable could not be used, because for the distance of more than 500 meters several repeaters would have been necessary, and, furthermore, there would have still remained the risk of a ground potential difference between the two distant places in the building.

An uplink to the internet was also installed for NTP time synchronisation and remote maintenance of the installation.

3 Recordings and Mixing

In the partner cities of Cologne, Rotterdam and Liège, a series of field recordings was carried out. The plan was to cover a certain amount of the typical urban presence of both cities. A set of categories for urban functions helped to structure the recordings according to the needs of the score rules.

For the recordings, various recording techniques and microphone types were used, such as

shotgun and M/S sets as well as binaural microphones, contact microphones (accelerometers) and underwater microphones (hydrophones).

The recorded material then was catalogued and used for mixing sound pieces of different kinds and lengths, as they were needed for the score. According to the facilities avaible at the Academy of Media Arts, mixing was done in a multichannel Protools studio using the prototype speakers of the installation. The last editing and mixing steps where done on site on a mobile Protools system, before the entire sound material was transferred to the Linux computer systems.

4 Score Rules

The aim when designing the score was to find a simple structure of the composition which would produce a big variety of possible arrangements, while still keeping certain parts of the process well recognisable by the listener.

The variety is achieved by using an algorithmic control with aleatoric selection processes for the tracks to be played next. However, full random control especially of combinations of files and of the spatialisation is also problematic, as it is difficult to assert a certain aesthetical level of the sounding result.

Therefore, a combination of more restricted and more open situations has been designed, which led to the definition of several score elements along with their functions.

4.1 Score Elements

4.1.1 Texts

The texts are mono tracks which were recorded with professional speakers, each text with a male and a female voice. Thus, the same text may appear in two forms.

In the score process, all 29 texts are formed to a series. This series is then played, one text after the other, but alternating in one and the other staircase. The next cycle would then repeat the same series, but starting in the other staircase and with the gender of the voices changed. After the second cycle, a new series would be formed and used for the next two cycles.

The pause between one text and the following text is varied each time, ranging from a pause of up to 24 seconds until –8 seconds, which would mean a short overlap of both texts. The bilinear probability distribution being used, however, concentrates the length of the pauses around zero seconds.

In the staircases, there are some loudspeakers reserved for text tracks. They are mounted at special places for an optimal speech comprehensibility. The spatial location of a text track is also varied according to an algorithmic series.

4.1.2 Klanginseln

Klanginseln ("sound isles") are used for combining them with the texts. Whenever a text track is running in one of the staircases, the other staircase would run a corresponding *Klanginsel* with the same length.

The correspondance to the text is formed by the urban categories mentioned in section 3. With every text, a set of three "allowed" categories (or category combinations) is associated, with decreasing probability of appearance. As the *Klanginseln* are also categorised, for every text an appropriate file can be algorithmically selected. There are 35 *Klanginsel* tracks for the Rotterdam staircase and 57 for Liège.

In order to always achieve the same playback length, *Klanginseln* are only used partially, or, in case the text is longer, are looped. Therefore, all *Klanginseln* have to be seemlessly loopable. Playback of a *Klanginsel* does not necessarily start from the beginning of the file. In fact, playback may start at any point in the file, which, over a longer period of time, would ensure that all parts of the *Klanginsel* appear equally often.

The spatial position of a *Klanginsel* in the staircase is also subject to algorithmic control. As described above, it was important to keep a certain level of sound projection quality independant from the position. Therefore, all *Klanginsel* tracks are mixed to two channel stereo files. In the routing of the playback system, several so called stereo *slots* would be selected randomly, which would then project the file onto one or more preconfigured speaker pairs. As opposed to the text loudspeakers, these stereo pairs are mostly mounted for diffuse sound radiation.

4.1.3 Trajekte

As opposed to a *Klanginsel* file, *Trajekte* ("trajectories") are sound compositions which use the full number of available channels, including the speakers reserved for text. They will be played back always in their full length, from the beginning to the end. They are neither associated with categories nor text tracks and stand for themselves.

Since a *Trajekt* is a static composition which

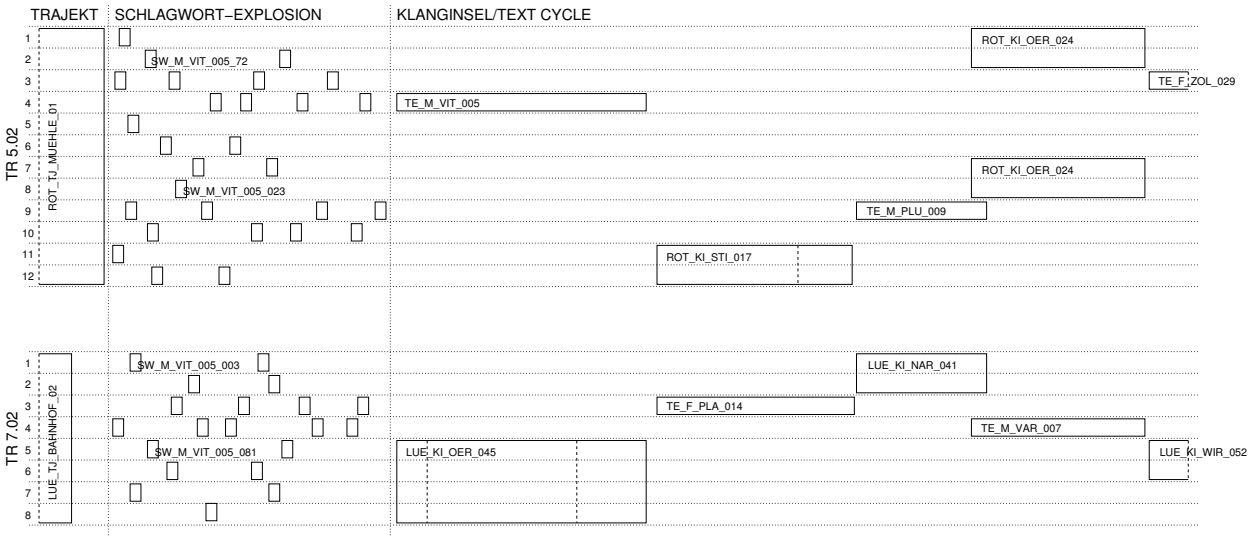

Figure 4: Score example of *GROMA*

is not further modified by the algorithmic control, it can make use of all spatial features possible in the staircase, such as travelling sound portions or the spatial distribution of sound elements according to their narrative function in the composition. For example, the "boat trip" *Trajekt* of Rotterdam harbour incorporates several vertical layers of recordings, from hydrophone under water recordings up to the topmost point of the boat, which are reflected in the spatialisation as well.

In both staircases, playback of *Trajekt* tracks would start simultaneously. Due to different lengths of the files and depending on the actual combination, one staircase would be finished earlier than the other. In this case, there will be an intentional pause in the composition, waiting for the other staircase to "become ready" again.

Every cycle of *Klanginsel/Text* combinations is followed by one *Trajekt*, before the next algorithmic cycle begins. Therefore, the installation constantly oscillates between these two poles: On one hand the algorithmic triggering according to a strict ruleset, which entangles the playback series of both staircases, but yields the more varying and unpredictable results; on the other hand fixed multichannel compositions, but with a more narrative dramaturgy and a more virtuosic way of dealing with space.

For each of the staircases, six *Trajekt* files were composed. They are also organised in a series, which will determine their playback order for the duration of one week. Every week, the series is rearranged.

4.1.4 Schlagwort-Explosion

A *Schlagwort* ("keyword") list is a separate version of each text, and consists only of some remarkable words or word groups appearing in the text. It is therefore a completely fragmented incarnation of the text, but still partially understandable through association, "human autocompletion" or prior knowledge of the text, for example by having listened to it already in the installation.

The *Schlagwort* lists were read and recorded separately, word by word, from both the male and the female speakers. Depending on the length of the text, there are up to several hundred *Schlagwort* files belonging to it.

Schlagwort files are used in the composition as a transition between the *Klanginsel/Text* and *Trajekt* cycles, and vice versa. After the last text in a cycle has finished, its *Schlagwort* list will start to play the fragments in their order of appearance in the underlying text. Gradually, the order will become more and more chaotic and increasingly random. At the same time, this process also affects the spatial positions: starting from the reserved text speakers, the fragments will be distributed more and more over all available channels. After a certain time, the process will arrive at full randomness. The fragments then stop and the *Trajekt* will start to play.

In the other case of leading back from a *Trajekt* to a *Klanginsel/Text* cycle, the fragments of the first text in the following cycle will be used. The development of randomness will be backwards in this case, starting completely chaotic

both in terms of order and spatialisation and going on towards the original order and more restricted to the text speakers.

This compositional element of using fragmented clouds of words was given the name *Schlagwort-Explosion*.[1] It exemplifies the two aesthetical poles of the entire composition in a subtle way, but also hints at fragmentation processes in history and urban development.

4.2 Example Score Incarnation

Figure 4 shows a synthetic, yet possible example incarnation of a score excerpt according to the rules described above. Both staircases are shown with their speaker channels, time is advancing from left to right. At the beginning, a *Trajekt* is running on all channels. In the second staircase, there is a pause due to a shorter file. Then there follows the *Schlagwort-Explosion*, followed by a *Klanginsel/Text* cycle. Typical possible situations are shown, such as looping a *Klanginsel* track twice, distributing a *Klanginsel* over two or four channels, and the gender change of the text tracks.

5 Implementation

5.1 Requirements

GROMA is a permanent sound installation. Therefore, its implementation has to be reliable and stable. In case of a power failure, it has to reboot automatically and continue to work seamlessly.

At the same time, *GROMA* is a site specific installation. Major parts of the score rules and almost all (fine) tuning settings could only be done on site. For most of the setup time of the installation in 2007 and 2008 this meant quasi-outdoor working in winter. In order to allow for a statisfying workflow of experiments and adjustments in a team of three artists, a high flexibility had to be maintained in the implementation.

For both reasons, a Linux audio distribution as the operation system and Supercollider as the sound framework were the tools of choice for *GROMA*.

5.2 Supercollider

Supercollider is a modern, versatile sound programming environment originally developed by James McCartney. It has been a commercial, closed source application for Macintosh computers only. In 2002 and starting with version 3, it became Open Source, Free Software, and was further developed by the community. This included a port of Supercollider to Linux, which was done by Stefan Kersten.

Supercollider 3 is designed after a client-server model. The server *scsynth* is a realtime audio synthesis engine, while the client *sclang* implements a programming language for controlling the server. *sclang* and *scsynth* communicate with Open Sound Control (OSC) [6].

5.2.1 scsynth

The Supercollider server follows a simple and modular, thus very powerful model. The actual sound producing or signal processing entities are called *synths*. They can be dynamically created and deleted at runtime. The layout of a synth is determined by a synth definition, or short *synthdef*, which describes an interconnected network of *unit generators* (*ugens*).

This model dates back to the *Music N* language family and is similar to that of Pd or Csound, where a synth would be called an *instrument* and a ugen an *opcode*.

Synths are reading from and writing to *buses*. Buses carry signals either with audio rate or control rate. Multiple buses may form a multichannel bus.

Synths can be combined to *groups*. All synths belonging to one group can be controlled together. The order of execution of the synths is also determined by the order of groups. Groups can be nested into other groups.

This design of *scsynth* allows for creating highly flexible, complex sound synthesis or signal processing networks.

All tasks on the server, such as synth and group creation and destruction, setting control values etc. are done with a set of OSC commands. Therefore, *scsynth* can be controlled by any application implementing these commands, not only *sclang*.

5.2.2 sclang

The Supercollider language is the standard client for the *scsynth* application. It implements a Smalltalk-inspired object oriented language with a C-style syntax and an incremental, realtime safe garbage collector. It also provides many features of functional programming languages, which are especially useful for algorithmic composition.

[1] The term *Schlagwort-Explosion* is derived from the radio play *Die Maschine* ("the machine") by French writer Georges Perec, where towards the end the fictious computer performs a *Zitatenexplosion* ("explosion of citations").

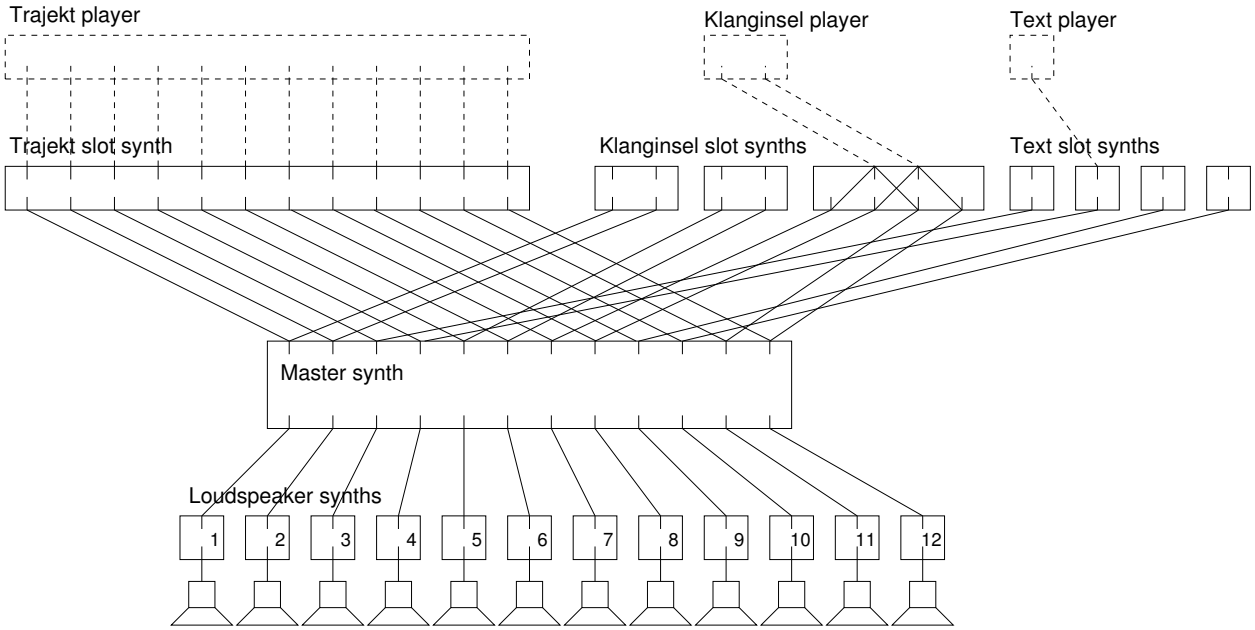

Figure 5: Routing layout of the *GROMA* system

5.3 Play *GROMA*

One major artistic decision for the score of *GROMA* was to only use preprocessed audio material as the score elements. Therefore, no realtime sound synthesis (in a strong sense) is involved.

The degrees of freedom for the realtime process are reduced to the synchronised selection of the sound material to be played according to the score rules, and the algorithmic control of the spatialisation. Thus, the *GROMA* system basically consists of a multichannel, dynamically automated file player with an advanced routing system, all built with Supercollider.

5.3.1 Routing

Figure 5 shows the basic routing structure in the *GROMA* system.

The signal flow starts at the player synths, which are dynamically created depending on which file is currently played back. Their outputs connect to so called slot buses or simply *slots*. Slots are mono or multichannel buses which correspond to the elements of the score. The *Trajekt* slot therefore has twelve or eight channels, depending on the staircase, while the *Klanginsel* slots have two channel stereo inputs. The number of *Klanginsel* slots also shows the multiple possible spatial locations of this score element. The corresponding synths do the routing two one or more output stereo pairs, as shown in the figure.

Likewise, there are text slots and *Schlagwort* slots (the latter have been omited in figure 5).

All slot synths output their signals to the master bus. They are read by the master synth, which does the global processing, such as fixed and dynamic global volume control.

On the output side of the master, the split mono signals are routed through one separate synth for each speaker, before they are sent to the sound card.

```
SynthDef('gr_speaker_std', {
    arg gain, out, in;

    Out.ar(out, gain * In.ar(in, 1));
}).writeDefFile;
```

Figure 6: Simple loudspeaker slot synthdef

This routing allows for the required flexibility when setting up the installation: changes can be made dynamically during runtime at every stage in the routing process. It is possible to fine tune the outputs on a per-speaker basis, which most of the time applies to volume, but as there are separate synths per speaker, they could also have a completely different layout for e. g. applying a filter correction.

On the other end, adjustments can be made to each of the score elements and to each input slot separately. A *Klanginsel* slot with a two channel stereo output most probably would need another gain factor than one playing to three stereo pairs. For some of the text slots, some slight deessing turned out to be necessary.

```
SynthDef('gr_slot_te_deesser', {
    arg gain, out, in,
      matrix = #[ 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0 ],
      preGain = 1.0, sideGain = 1.0,
      sideFreq = 5000, compThresh = 0.5,
      compRatio = 0.25, compAttack = 0.002,
      compRelease = 0.01;

    Out.ar(out, matrix * Compander.ar(
      in: preGain * In.ar(in, 1),
      control: HPF.ar(In.ar(in, 1),
        sideFreq, sideGain),
      thresh: compThresh, slopeBelow: 1.0,
      slopeAbove: compRatio,
      clampTime: compAttack,
      relaxTime: compRelease,
      mul: gain));
}).writeDefFile;
```

Figure 7: Text slot synthdef with deesser

The `GrRouting` class allows for comfortable runtime creation and modification of all the synths in the routing. A synth may be replaced by one with a different underlying synthdef configuration, or the settings of one synth including its synthdef may be applied to another synth. For the deessing example above, this mechanism served for using the experimentally determined values for the first text slot as a basic setting for the other text slots.

For implementing this convenient functionality, some pitfalls had to be taken into account.

A running synth's definition actually cannot be simply changed, rather the old synth has to be deleted and a new synth using the new synthdef has to be created.

Depending on their function in the routing, synthdefs in *GROMA* have some standard control parameters, such as `out` and `in` fields or multichannel output matrices, whose values form the actual routing layout. Additionally, there may be "private" parameters to some of the synths, such as filter parameters.

In order to keep the routing layout intact when replacing a synth, the routing information of an old synth has to be kept. In `GrRouting`, the standard control values of the old synth are therefore queried from the server before deleting it. They are then applied to the new synth after its instantiation.

When copying the parameters of one synth to another, the routing channel information of the target synth has to be kept, while the new synthdef and additional "private" parameters need to be applied. This is handled automatically in `GrRouting` by a careful layout of all the synthdefs and a defined number of standard controls per synth class.

5.3.2 Saving the Routing Layout

When the routing of the *GROMA* system is dynamically adjusted and tuned, the actual state of the routing is only present on the Supercollider server. When the server is quit, all the settings are lost. Therefore, the complete routing settings need to be saved and reapplied on restart or reboot.

One solution to this would be to track all the client-side changes to the routing during the setup period. The downside of this approach is that it reduces the possible ways of manipulating the routing. For the setup procedure of *GROMA* it was desirable to keep the full flexibility for making adjustments, such as using the "low level" *sclang* methods for controlling synths, convenience methods of `GrRouting`, custom functions, or even an external remote application sending OSC commands directly to *scsynth*, which would be hard to track.

Therefore, a separate class `GrSynthConfig` has been implemented, which traverses the synth graph of `GrRouting`, queries the definition and the control vector of all synths, and saves them to a configuration file. In the opposite direction, the values of the configuration are applied to the routing. This could mean to populate a completely empty `GrRouting`, e. g. at system startup, or to apply only the differences of the configuration to the current routing state. This allows for easily trying out different configurations and for a kind of "undo" function in case of misconfiguration.

5.3.3 Play Sound Material

All sound content of *GROMA* has to be played back directly from disk, as the overall size of the sound material would be over 55 GiB in 32 bit floats.

In Supercollider, the `DiskIn` ugen streams audio data from disk. For the seemless loop playback of *Klanginsel* files, however, `DiskIn` had to be extended with a loop functionality, which starting from version 3.2 was added to the standard distribution of Supercollider.

The class `GrTrack` provides the client side means for instantiating a player synth. As `DiskIn` does not provide a way to signal the end of a file back to the client, an envelope is applied to all files, which, after completion, deletes the

```
SynthDef('gr_play_tj12_std', {
    arg gain, out, bufnum,
      env = #[0, 0, 0], loop = 0;

    Out.ar(out, EnvGen.ar(Env.linen(
        env[0], env[1], env[2], gain),
      1.0, doneAction: 2) *
      DiskIn.ar(12, bufnum, loop));
}).writeDefFile;
```

Figure 8: Player synthdef for *Trajekt*

enclosing synth. For *Klanginsel* tracks, an envelope is necessary anyway, since it is looped until the corresponding text track is finished.

Another class, `GrPlay`, forms the high level access to the tracks. It uses a database of all sound material of the installation in order to lookup the filename, a per-file gain factor, and the score element class with its set of possible playback slots for a given sound name.

The algorithmic score control accesses only this single function to trigger the complete score unfolding of *GROMA*.

```
play { arg name, slot = 0, start = 0.0,
    env = nil, duration = nil,
    fadeTime = 0.1, loop = 0, run = true,
    action = nil;
```

Figure 9: Head of the `play` method in `GrPlay`

5.4 Score Implementation

5.4.1 Algorithmic Principles

Supercollider provides excellent functionality for implementing the score rules described in section 4. One important concept is the mechanism of *Patterns* and *Streams*.

A pattern is the template for forming a stream of values, which then can be accessed one by one with sending the `next` message to the stream object. The following code creates a stream with a random order of the numbers from 1 to 10, which will repeat infinitely, and queries the first value:

```
p = Pshuf.new((1 .. 10), inf);
s = p.asStream;
s.next;
```

Since Patterns can deal with objects of any class as their values, the series of text and *Trajekt* tracks in *GROMA* could be realised simply using this mechanism.

Another task when implementing the *GROMA* score rules are random selections of tracks out of collections, such as arrays.

```
[ 1, 2, 3 ].choose;
```

would select one of the elements of the array with equal distribution, while

```
[ 1, 2, 3 ].wchoose([ 0.6, 0.3, 0.1 ]);
```

would do the selection based on the probability weights for each element given as an argument.

The lookup of *Klanginsel* tracks for a given text track is implemented as a combination of a weighted and an evenly distributed `choose` operation on the database of category associations, which have been described in section 4.1.2.

The *Schlagwort-Explosion* uses both techniques: there are two streams, one for the ordered sequence of *Schlagwort* fragments of a text, the second for the shuffled sequence. A weighted `wchoose` operation on an array of both streams with an increasing or decreasing weight realises the transition from the ordered to the chaotic appearance of the fragments. The same mechanism is used for the transition of spatialisation from the reserved text speakers towards all speakers.

5.4.2 Realtime Score vs. Pregenerated Score

The parts of the *GROMA* score were prototyped in a realtime version in order to test and adjust the score rules for the installation. The selection principles take place and immediately trigger the playback using the `GrPlay` class.

For the final production system, however, some issues would need to be solved when using a realtime score.

First, the random seeds of the Supercollider threads' random number generators would need to be saved regularly in order to be able to seemlessly continue with the random sequence after a power failure or system crash.

Second, the synchronisation between the two staircases would be a non-trivial task. One possible solution would be to elect one machine the "score master", which then controls both staircases. In case of a network or power failure, the slave would need to dynamically become a master. When the network connection is back, the master would need to be negotiated again. For all serial operations, such as *Trajekt* and text sequences, the slave would need to track any state change in order to be able to seamlessly continue with the score in case the master disappears.

Although there are standard solutions to all of these problems and although it is possible to implement these in Supercollider, the much easier approach of a pregenerated score was chosen.

Several decades of score data have been generated in advance and stored on both machines. Each score event entry includes an absolute point in time for its playback. Thus, both machines are completely independant from each other and just need to be synchronised to a global NTP time server. The absolute time values of all score events ensures the exact synchronicity of the distributed score in both staircases.

As the uncompressed score data size is approximately 1.5 Gigabytes per decade, the data was split into several files which are loaded incrementally into memory on demand.

Of course, the decision of using a pregenerated score might be questioned as a conceptual renunciation from the "eternal realtime composition" *GROMA* originally aimed at. On the other hand, the audible result is exactly the same in both cases. The fact that hardware failures, vandalism, high water, construction changes or destruction of the buildings even to the point of war, however, are much more likely to harm the eternity of *GROMA* than running out of pregenerated score data opens an interesting area of thought.

6 Conclusions

GROMA is a permanent urban sound installation, being an intrinsic part of the new *Rheinauhafen* neighbourhood of the city of Cologne. In this paper, it has been described how urban sound situations from two partner cities of Cologne, Rotterdam and Liège, are algorithmically combined with ancient texts. The aesthetical considerations and the artistic process towards a rule set for the score of the installation have been exemplified. The technical details of the realisation of *GROMA* and the balance between technically and artistically motivated decisions have been shown.

7 Acknowledgements

My thanks go to my team members Michael Scholz, who is the initiator of this project, and Judith Nordbrock, sound engineer at the Academy of Media Arts.

The team's further thanks go to the project partners HGK Häfen und Güterverkehr Köln, modernes köln Stadtentwicklung GmbH, Fraunhofer Institute IAIS, Johannes Heppenheimer (Audiance), Netcologne, Manfred Fox Engineering Berlin, Anselm Görtz (Audio & Acoustics Consulting Aachen), Astrid Lutz (arts+org), and Prof. Klaus Schöning.

All photos taken by Michael Scholz.

References

[1] Michael Scholz, Judith Nordbrock, and Martin Rumori. Website of GROMA, 2008. Available from World Wide Web: `http://www.groma-net.de` [cited 2009–01–28].

[2] Häfen und Güterverkehr Köln AG. Rheinauhafen, 2008. Available from World Wide Web: `http://www.rheinauhafen-koeln.de` [cited 2009–01–28].

[3] Johannes Heppenheimer. Audiance, 2008. Available from World Wide Web: `http://www.audiance.net` [cited 2009–01–28].

[4] 64 Studio Ltd. 64 studio, 2008. Available from World Wide Web: `http://www.64studio.com` [cited 2009–01–28].

[5] James McCartney and et al. Homepage of supercollider, 2008. Available from World Wide Web: `http://supercollider.sourceforge.net` [cited 2009–01–28].

[6] Open sound control protocol, 2008. Available from World Wide Web: `http://www.opensoundcontrol.org` [cited 2009–01–28].